Towards Better Scalability for IoT-Cloud Interactions via Combined Exploitation of MQTT and CoAP

Paolo Bellavista, Alessandro Zanni
Dept. Computer Science Engineering (DISI)
University of Bologna, Italy
{paolo.bellavista, alessandro.zanni3}@unibo.it

***Abstract*—It is manifest the growing research and industrial interest in scalable solutions for the efficient integration of large amounts of deployed sensors and actuators (Internet of Things – IoT- devices) and cloud-hosted virtualized resources for elastic storage and processing (including big data online stream processing). Such relevant attention is also demonstrated by the emergence of interesting IoT-cloud platforms from industry and open-source communities, as well as by the flourishing research area of fog/edge computing, where decentralized virtual resources at edge nodes can support enhanced scalability and reduced latency via locality-based optimizations. In this perspective, this paper proposes an innovative distributed architecture combining machine-to-machine industry-mature protocols (i.e., MQTT and CoAP) in an original way to enhance the scalability of gateways for the efficient IoT-cloud integration. In addition, the paper presents how we have applied the approach in the practical experience of efficiently and effectively extending the implementation of the open-source gateway that is available in the industry-oriented Kura framework for IoT.**

Keywords—Internet of Things; Scalability; Cloud-Sensor Interaction; Gateway; MQTT; CoAP.

I. Introduction and Paper Contributions

In the latest years, the IoT perspective has become more and more relevant, as well as the number of sensors/actuators involved in IoT scenarios of practical industrial interest has rapidly increased, see for instance the smart city application domain. In addition, as many related research activities are showing, an exponential growth of the number of IoT devices considered in realistic applications will continue in the near future [1]. Therefore, scalability is emerging as one of the key factors for IoT development and exploitation, also considering the technical challenges connected with possible geographical distribution over wide areas, connectivity through federated networks and resources in general, as well as the need to enhance the applicability of traditional security, trust, and privacy management infrastructures in an IoT-specific and efficient way [2].

Even if recognized as a central technical challenge in the field, research on IoT scalability is still in its infancy, with many existing IoT frameworks (targeting both the academic and industrial communities) that still do not address the scalability requirements properly and adequately. Among the recent research solutions in the field, it is worthwhile mentioning [3] that addresses the technical issue of lack of appropriate scalability in existing trust management protocols and [4] that integrates smart resource-constrained objects into the Internet by using virtual networks, instead of gateways, by providing end-to-end communication between devices. In addition, [5] outlines a rule-based intelligent gateway that bridges the gap between existing IoT protocols to enable the efficient integration of horizontal IoT services. [6] proposes a powerful IoT gateway based on a customizable architecture to be attached to different networks and adapt to different application requirements. [7] provides translation mechanisms to communicate with sensor networks, in order to create a universal infrastructure connected with IoT gateways. Even if often oriented to adopt gateway-based architectural solutions, the above approaches do not fully exploit the potential of gateways to significantly improve scalability while remaining compliant with industrially accepted standard protocols in the field, such as Message Queue Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP), in order to promote wide adoption and easy integration with existing deployment environments.

Within this general perspective, this paper describes our original research work of design and implementation of a scalable distributed architecture for the dynamic management of IoT resource interactions with the cloud. Our solution extends (and has been designed and implemented on top of) the Kura framework, a platform for IoT-cloud integration of strong industrial relevance, based on Java OSGi and on the role of infrastructure gateways for efficient message brokering [8-11]. In particular, we have extended the Kura framework to offer improved scalability and reduced latency for communication/coordination among wide-scale sets of geographically distributed IoT devices interworking via gateways. Based on a dynamic tree structure organization, our gateways and IoT devices exploit MQTT and CoAP in a combined way to improve the efficiency of i) node communications for efficient resource look-up; ii) IoT device discoverability; iii) resilience to device disappearance or unexpected disconnection; and iv) dynamic and lightweight hierarchy reconfiguration, triggered by ii) and iii).

In particular, about compliance with industrially accepted machine-to-machine protocols, we propose a significant Kura extension that integrates CoAP in order to enhance Kura functions and to optimize node management via our hierarchical tree organization, with the primary goal of improved scalability. In fact, MQTT plays a central role and is strongly exploited by the Kura framework, but is exhibiting non-negligible limitations in terms of scalability, e.g., inefficient usage of TCP con-

978-1-5090-1131-5/16/$31.00 ©2016 IEEE

nections towards the broker when growing the number of IoT devices in a gateway locality. Our original idea is to extend the MQTT-oriented Kura gateway with more lightweight CoAP-based coordination functionality, thus achieving scalable interactions, especially in the challenging case of node/resource discovery. Moreover, we claim that the proposed solution is strongly original in several technical aspects:

- it is built on top of an impactful and widely used framework, i.e., Kura, by taking advantage of several efficient and state-of-the-art technologies, such as Guava and Kryo;
- we design and implement a potentially impactful extension of the Kura gateway, based on an original, dynamic, and efficient tree structure for inter-node communication and coordination;
- as a side effect of our innovative research work, we are able to qualitatively and quantitatively evaluate MQTT and CoAP, in particular in terms of performance and scalability;
- it exploits the combined and synergic integration of emerging standard protocols of wide industrial interest, i.e., MQTT and CoAP, which are recognized to be generally lightweight, flexible, asynchronous, and secure (when efficiently integrated with SSL and DTLS mechanisms).

Let us note that some gateway-oriented solutions for enhanced IoT scalability (not jointly exploiting the MQTT and CoAP protocols) have already been published in the recent literature in the field, thus showing the general suitability of the gateway-based architectural approach. [12] presents a system architecture for scalable IoT cloud services that benefits from CoAP adoption: it is shown to be able to outperform HTTP Web servers and to achieve good throughput in constrained environments. [13] proposes a scalable architecture based on CoAP for service and resource discovery in large-scale IoT applications via proxying and caching. [14] proposes a lightweight IoT-oriented home gateway based on MQTT, by offering device discovery/management primarily oriented to the integration with third-party and legacy platforms. Finally, [15] proposes a lightweight gateway to manage large amounts of endpoints, based on a collection of Web Services components that send/receive sensors and actuators data using the COnstrained Restful Environments (CORE) link format (https://tools.ietf.org/html/rfc6690).

To the best of our knowledge, there is no solution yet in the related literature that has adopted a gateway-oriented architecture where gateways jointly exploit MQTT and CoAP to achieve highly scalable IoT device management through dynamic hierarchical tree organizations.

In the remainder of the paper we outline the needed background about Kura, MQTT, and CoAP, in order to fully understand the technical originality of our proposal in Section II. Section III and Section IV respectively present the primary design guidelines and technical insights about our efficient implementation of the Kura gateway extension. Some performance analysis, related discussion, and conclusive remarks will end the paper.

II. Technology Background

This section rapidly overviews the primary technological background needed to fully and easily understand our following original proposal, which is based on the exploitation of the Kura framework, MQTT, and CoAP.

A. Kura Framework

Kura [16] is a framework for IoT applications that also provides a platform for building IoT gateways. In particular, it abstracts the design and implementation of gateways from the complexity of real-world industrial scenarios consisting of heterogeneous hardware/network devices. To this purpose, Kura aggregates and controls device information and supports the simplification of the overall development and deployment process. Kura is based on Java OSGi to support, in a widely accepted way, the dynamic management of software components with no need of operation suspension. It simplifies the process of implementing reusable software building blocks and of creating self-contained pluggable packages (i.e., bundles), specifically suitable for IoT applications. In particular, as detailed below, we have decided to base our implementation on Kura because it can serve as a suitable container for Machine-to-Machine (M2M) applications running in service gateways and can employ a large set of network protocols to communicate with lower-layer devices. In addition, Kura has the capability to support Virtual Private Networking (VPN), firewalls, and Network Address Translation (NAT) traversal. Finally, Kura is open-source with a fervent and growing community supporting the project, continuously proposing general-purpose and application-specific extensions.

B. MQTT

MQTT [17] is a lightweight message-oriented protocol following the publish/subscribe model. It allows achieving good scalability and can dynamically support a wide range of applications, especially in the IoT and M2M domains. Every MQTT resource is modeled as a client and can connect to an MQTT broker over TCP. MQTT has intrinsic characteristics that make it a valuable option in IoT environments with low-latency, low-bandwidth, and power efficiency requirements, e.g. small header overhead, topic-oriented management, and automatic message forwarding when clients reconnect. Moreover, it provides reliability with the possibility to dynamically select one of three QoS level options for message delivery:

- QoS0 - At most once semantics, with the best-effort mode of the network. The delivery is not acknowledged and the message is not stored. The message could be lost or duplicate. It is the fastest mode of transferring messages;
- QoS1 - At least once. Each message might be delivered multiple times, with possible duplications, if failures occur before an acknowledgment is received by the sender. With this option, messages must be stored locally at the sender, until they have been delivered to their receiver, so to enable possible re-transmissions;
- QoS2 - Exactly once, with the guarantee that no duplication of messages occurs. It extends QoS1, by storing messages also at receivers in order to avoid any duplications.

978-1-5090-1131-5/16/$31.00 ©2016 IEEE

Finally, MQTT allows every node to register a broker-side message (Last Will and Testament) that the broker sends to all subscribed clients on the topic, when the node disconnects unexpectedly. This provides a basic automated mechanism for the monitoring and management of disconnections in highly dynamic IoT environments.

C. CoAP

CoAP [18] is a document-transfer protocol optimized for M2M communications between constrained devices. CoAP provides a request/response interaction model, exchanging small messages on the UDP transport layer. It is suitable for constrained nodes and constrained (e.g., low-power, lossy) networks: it uses a short fixed-length binary header of 4 bytes, possibly followed by compact binary options and a payload [19]. In particular, CoAP allows i) adding UDP support for unicast and multicast requests with optional reliability, ii) exchanging asynchronous connection-less datagrams, iii) low header overhead on packets, iv) low complexity of parsing operations, and v) supporting simple mechanisms for proxy and caching actions. About reliability, CoAP supports four options in terms of different types of messages, namely Confirmable, Non-confirmable, Acknowledgement, Reset.

III. The Design of an original Kura extension for scalability through CoAP-based Hierarchical Coordination

As already stated, our original proposal is based on the integration and the synergic interworking of the MQTT (already integrated into Kura) and CoAP protocols in order to improve overall scalability. Both protocols are open, lightweight, and typically used in constrained environments.

In particular, we have decided to use MQTT for inter-node communications inside the hierarchical tree organization of IoT devices that our solution dynamically organizes to efficiently manage large scale via hierarchy- and locality-oriented optimizations. We exploit MQTT for communications between CoAP servers in order to retrieve resource information and for node synchronization in our tree management procedures. The MQTT publish/subscribe model allows reaching multiple nodes by limiting the overall number of exchanged messages, thus achieving good performance also when our hierarchy consists of a large number of nodes. Publishers and subscribers are completely decoupled from each other thanks to the usage of the MQTT broker, thus simplifying the addition of new nodes and making any node able to seamlessly interact with the others. In our solution, each participant node can publish messages independently to the receiver nodes' state and the broker may send messages to nodes when they turn active; also in the case a subscriber is performing a non-interruptible operation, the broker can queue messages to be processed later (temporal decoupling). Moreover, nodes can exchange messages without any prior knowledge of each other (spatial decoupling).

Based on our tree structure (see below), MQTT uses hierarchical topics. Nodes may subscribe to a topic and then observe the whole hierarchy by using wildcards. In addition, MQTT provides communication reliability, if needed, by enabling secure transmission for hierarchy control messages: reliable communications exploit TCP and differentiated QoS levels. The TCP handshake to the public broker address and the persistent connection establishment allow reaching hosts behind NAT without problems, a common configuration when dealing with real IoT devices in many practical deployment environments.

As already stated, notwithstanding its many pros, it starts to be recognized that MQTT also presents some drawbacks in its pervasive IoT employment. Only to mention a first notable example, being based on TCP, it is more suitable for resource-full nodes than for constrained IoT devices (e.g., consider the multi-step handshake needed every time new nodes are establishing a TCP connection). Moreover, TCP connections are kept open, with the management of a persistent node-broker session, thus often generating useless resource consumption, in particular at IoT endpoints.

To partially solve the above issues, a possible solution may involve to set the MQTT clean session flag on the broker to choose to either keep the connection open or remove it after the transmission. Unfortunately this approach is not usually viable in practical IoT environments because not using TCP persistent sockets means having to re-establish connections through the multi-step handshake every time an endpoint sends data. That would lead to unacceptable performance decrease in several application scenarios (more data traffic, decrease of battery-operated life, also given the usual high sample rate of IoT endpoints).

Therefore, for multiple motivations, we believe that an MQTT standalone exploitation is not enough; in addition, we claim that CoAP is the right protocol to complement MQTT in order to overcome its limitations when applied to large-scale IoT deployment scenarios. In particular, for IoT endpoints locally connected to a node, MQTT introduces overhead for functions that are not crucial: using a less reliable but also less resource-demanding connection is often more suitable, especially if the sporadic loss of partial data is not critically impacting the overall system behavior, as it regularly occurs for cyber-physical monitoring scenarios. CoAP works asynchronously over UDP, which is connection-less, with higher performance, smaller packets, and reduced overhead. In fact, it starts to be recognized that CoAP characteristics make it specifically suitable for low-energy consumption application domains [19].

In particular, by following the seminal work in [20], we exploit CoAP to easily enable IoT devices to be discovered and to expose them externally as resources accessible with REST-call methods, using the */.well-known/core* URIs [21] with resource descriptions in the CoRE link format. By delving into finer technical details, each CoAP endpoint has associated the CoRE link attributes, which extend those in Web Linking [22], e.g. resource type ("rt"), interface description ("if"), content-type ("ct"), maximum estimated size ("sz") [23]. This allows clients to discover which resources are provided and their associated information before accessing them.

We employ CoAP for interactions where we need direct and very responsive lightweight communications, with low reliability constraints, i.e., between the CoAP server and the IoT endpoints due to the high data exchange rate between

978-1-5090-1131-5/16/$31.00 ©2016 IEEE

them. CoAP allows achieving these goals in single localities, where NAT drawbacks are ineffective and communication reliability is anyway adequate. Vice versa, although CoAP may be a standard-base for scalable architecture, as already stated, it has some limitations stemming from UDP usage, lack of advanced quality support, and NAT tunneling and port forwarding. In addition, [1] underlines the unsuitableness of using CoAP alone because it does not allow to aggregate IoT resources into a hierarchical organization. For these reasons, we have decided to integrate a CoAP-based solution with MQTT, as detailed in the following parts of the paper, thus complementing the functions and strengths of both protocols.

In particular, our CoAP-based gateway-oriented distributed architecture is sustained by five original support services, each one implemented in terms of an OSGi bundle. Every node is suited by the Kura framework that executes the bundles. The service base bundle is used to provide libraries to external bundles or to retrieve information about network addresses currently used by the node. The bundle contains a CoAP server, using UDP to have full access to the whole functionality of the open-source and widely adopted CoAP Californium framework. It retrieves data from either local resources or remotely through the Remote Query Resource (RQ) service, and uses a Resource Directory (RD) data structure to store all the information about resources.

Externally, each node is connected with the others in a tree-structure hierarchy. The tree structure consists of multiple OSGi bundles replicated in each node (with the associated RDs) and is based on two main components: CoAPTree-Handler (CTH) that manages all the CoAP servers hierarchy; MQTT broker, to exchange both resource and control messages to synchronize CoAP servers. Figure 1 depicts our high-level architecture proposal.

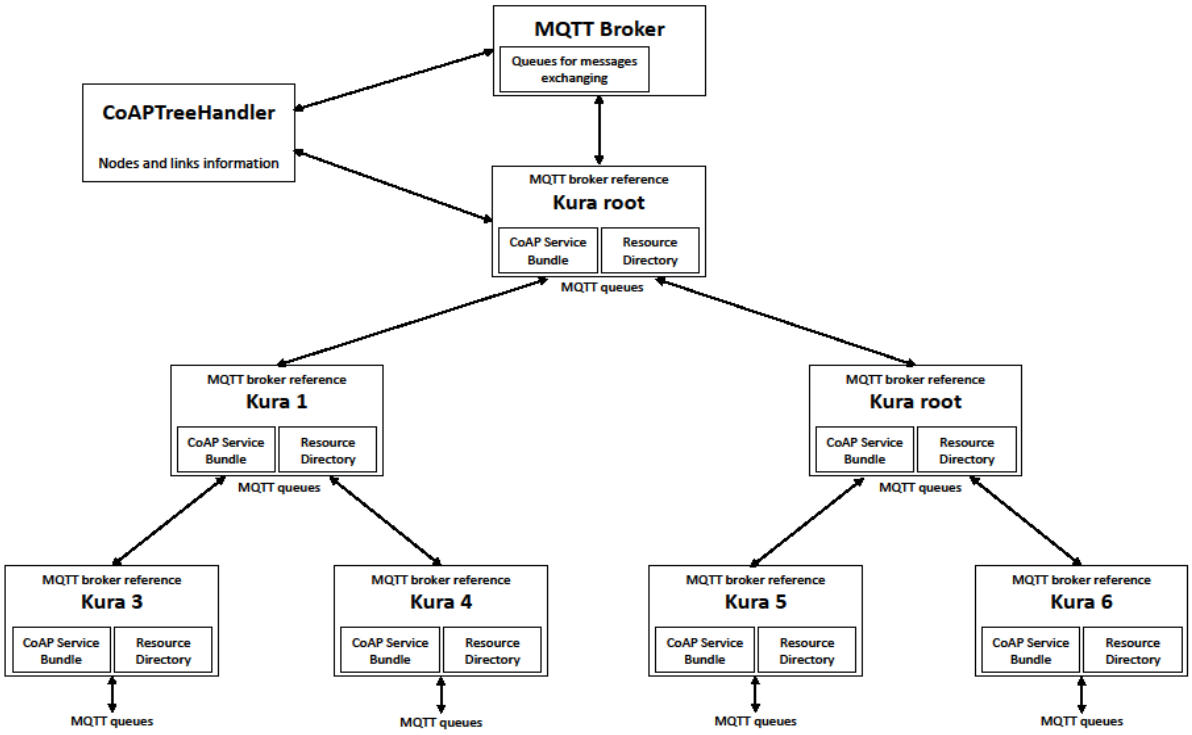

Fig. 1. The proposed architecture integrating MQTT and CoAP for better scalability.

Our CoAP servers need to exchange messages for resource requests and sometimes, if needed, they update the hierarchy knowledge by disseminating information to all involved nodes. We introduce a multi-layers hierarchy and each node specifies a domain name and a group (or sub-group if necessary) name that identify the node into the hierarchy. Thus, it is possible to limit the interest towards external resources to store into the RD, e.g., only to a specific sub-group or domain, thus making the message exchange between brokers more efficient. As depicted in Figure 2, our hierarchy is divided into three levels:

- Level0 includes only the root node and enables inter-localities communications. Every message sent to a different locality passes through the root node;
- Level 1includes all the nodes belonging to a specific locality. It permits quicker update than level2 and receive updates directly from CTH. If the root node is added/removed, level1 updates level2 nodes when they sub-

978-1-5090-1131-5/16/$31.00 ©2016 IEEE

scribe to the related topics, without any private additional communication by CTH;

- Level2 includes all the nodes of a given locality and of a given group. It might also be divided into specific sub-groups. It has less temporal constraints than the other levels and, in case of parent disconnection, it receives periodically updates by level1 with a private communication.

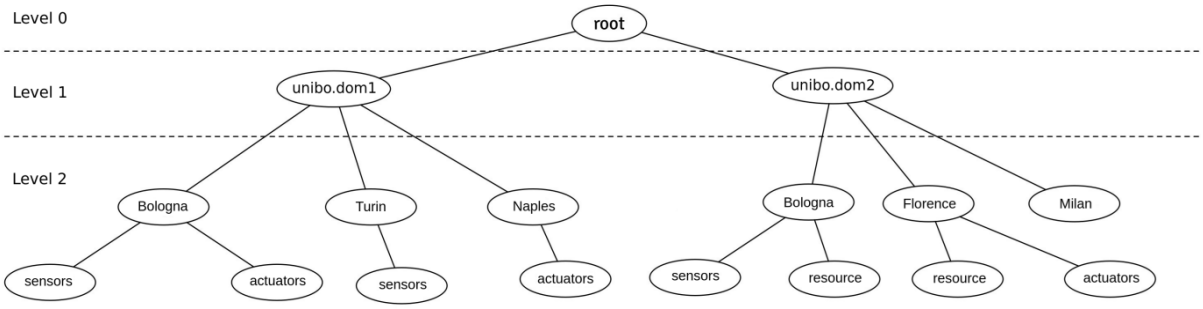

Fig. 2. The hierarchical tree structure of our extended Kura gateways.

IV. Some Implementation Insights

In this section we give some primary and central implementation insights about our tree-based resource management solution. In particular, we go into some in-depth technical details about the implementation of our internal node structure, replicated into all nodes, and of the tree-based hierarchy management components that connect the different nodes in our dynamic hierarchy.

A. Node Internal Structure

The internal node structure consists of multiple bundles, mainly for the sake of modularity and loose coupling:

- CoAP Server bundle. It is dedicated to the management of all the properties of registered resources and can be coordinated with other servers to perform callback methods in relation to the received messages. It works locally on a node to create a high-performance CoAP support inside Kura, thus enabling the possibility to send CoAP REST requests to discoverable devices;
- Resource Discovery (RD) bundle. It is a data structure used to store endpoints and resources belonging to different domains, groups, or subgroups. It supports usual discovery operations, e.g., to register, maintain, lookup, remove endpoints and resource descriptions. It includes an automated support to node disconnection management: if an endpoint does not update the RD periodically, it is automatically removed;
- Remote Query resource (RQ) bundle. It supports resource lookup by CoAP servers in our hierarchy, thus allowing nodes to retrieve information from CoAP servers by interfacing CoAP requests with MQTT brokers. RQ and CoAP servers are completely decoupled to enable servers to call RQ only in case of queries on remote resources; for local resources, they can exploit optimized and direct CoAP requests with no RQ interaction;
- CoAP Resource bundle. It is a service dedicated to the external exposition of discoverable resources. It can add, delete, or modify the exposed attributes.

B. Hierarchy Management

The management of our hierarchical tree is performed by:

- MQTT Broker bundle. It is a simple extension of a regular MQTT broker to exchange messages between nodes within our hierarchy; we have implemented it based on Mosquitto [24];
- CoAPTreeHandler bundle. It receives requests from Kura nodes and returns references to create our tree structure. It can perform different management operations, e.g., handling node replicas, providing hierarchy metadata, handling children nodes in case of parent disconnections. It is implement in an event-oriented way, where event sources are queue listeners to get updates from MQTT brokers. In case of failure of its instance, the CoAPTreeHandler is automatically replaced by a new instance according to a regular master-slave replication style.

V. Performance Evaluation

To assess and validate the feasibility of our approach, we have performed some tests on our tree hierarchy by using the deployment configuration of Figure 1. About the considered nodes, we have used RaspberryPi model B+ and Parallel VM (512MB RAM, single core CPU, Xubuntu OS); the VM hosts Kura with the org.eclipse.kura.emulator bundle. In particular, we have evaluated the performance of the two main critical scalability elements of our solution: the number of MQTT messages exchanged and the POST requests performed on RD. In fact, every time a node needs to communicate, e.g., to manage our tree hierarchy or to retrieve an endpoint attribute, our solution exploits MQTT messages, whose transport is natively integrated in Kura. The number of MQTT messages has demonstrated to depend linearly on both the number of levels and the number of nodes in our hierarchy. The number of CoAP requests to RD is generally high as well, due to the central role of RD in our architecture for intra-node communica-

978-1-5090-1131-5/16/$31.00 ©2016 IEEE

tions and endpoint attribute retrieval.

MQTT and RD evaluation tests have been performed by sending growing numbers of sequential MQTT requests and CoAP POST ones, with the purpose of evaluating our hierarchical solution under different load conditions. Table 1 and Table 2 report summarized results about the in-the-field measurements of MQTT and RD performance, respectively. In particular, the tables report the total time needed to complete the all messages transmission (average values after 100 runs). Let us note that, in both cases, our solution preserves scalability also in high-load scenarios: the total time shows to be proportional to the number of MQTT/CoAP requests with a constant and limited overhead, independent of load conditions.

Sender	Receiver	MQTT Messages	Time (s)
RaspberryPi	RaspberryPi	1000	14
VM	RaspberryPi	1000	5
VM	VM	1000	1
RaspberryPi	RaspberryPi	10000	120
VM	RaspberryPi	10000	60
VM	VM	10000	7
RaspberryPi	RaspberryPi	60000	723
VM	RaspberryPi	60000	377
VM	VM	60000	34

Table 1. MQTT performance results

Sender	Receiver	CoAP Requests	Time (s)
RaspberryPi	RaspberryPi	500	19
VM	RaspberryPi	500	4
VM	VM	500	0.8
RaspberryPi	RaspberryPi	1000	51
VM	RaspberryPi	1000	11
VM	VM	1000	1.8
RaspberryPi	RaspberryPi	5000	255
VM	RaspberryPi	5000	54
VM	VM	5000	3.3
RaspberryPi	RaspberryPi	10000	491
VM	RaspberryPi	10000	104
VM	VM	10000	6.9

Table 2. RD performance results

VI. Conclusive Remarks

This paper has originally proposed a flexible architecture for the IoT that scalably supports dynamic management and resource provisioning. The reported results show that the proposed solution, thanks to MQTT and CoAP interworking, can guarantee scalable support also for constrained devices, by preserving high message delivery rate also in busy network conditions, without generating operation anomalies (e.g., request timeouts or servers stopped due to overload).

Finally, given the promising results already achieved, we are currently working on i) further resource usage optimization in terms of CPU and memory usage (e.g., reducing the number of simultaneously active threads for resource management) and ii) improved security support via integration with DTLS.

References

[1] L. Rodrigues, J. Guerreiro, N. Correia, “RELOAD/CoAP Architecture with Resource Aggregation/Disaggregation Service”, 2016.

[2] R. Roman, P. Najera, J. Lopez, “Securing the Internet of Things”, Computer, vol. 44, no. 9, pp. 51-58, 2011.

[3] F. Bao, I.-R. Chen, J. Guo, “Scalable, adaptive and survivable trust management for community of interest based Internet of Things systems”, 11th IEEE Int. Symp. Autonomous Decentralized Systems (ISADS), pp. 1-7, 2013.

[4] I. Ishaq, J. Hoebeke, I. Moerman, P. Demeester, “Internet of Things Virtual Networks: Bringing Network Virtualization to Resource-Constrained Devices”, IEEE International Conference on Green Computing and Communications, pp. 293-300, 2012.

[5] A.Al-Fuqaha, et al., “Towards Better Horizonal Integration among IoT Services”, IEEE Communications Magazine, vol. 53, no. 9, pp. 72-79, 2015.

[6] S.M. Kim, H.S. Choi, W.S. Rhee, “IoT Home Gateway for Auto-Configuration and Management of MQTT Devices”, IEEE Conference on Wireless Sensors, 2015.

[7] L. Wu, Y. Xu, C. Xu, and F. Wang, “Plug-configure-play service-oriented gateway-for fast and easy sensor network application development”, SENSORNETS, pp. 53-58, 2013.

[8] The Kura IoT Platform – https://eclipse.org/kura

[9] P. Bellavista, A. Corradi, C. Giannelli, "Mobile Proxies for Proactive Buffering in Wireless Internet Multimedia Streaming", 25th IEEE Int. Conf. Distributed Computing Systems Workshops, 2005.

[10] A. Toninelli, et al. "Supporting Context Awareness in Smart Environments: a Scalable Approach to Information Interoperability", ACM Int. Workshop Middleware for Pervasive Mobile and Embedded Computing, 2009.

[11] P. Bellavista, A. Corradi, C. Giannelli, "The Real Ad-hoc Multi-hop Peer-to-peer (RAMP) Middleware: an Easy-to-use Support for Spontaneous Networking", IEEE Symp. Computers and Communications (ISCC), 2010.

[12] M. Kovatsch, M. Lanter, Z. Shelby, "Californium: Scalable Cloud Services for the Internet of Things with CoAP", IEEE Int. Conf. Internet of Things, pp. 1-6, 2014.

[13] S. Cirani, et al., “A Scalable and Self-Configuring Architecture for Service Discovery in the Internet of Things”, IEEE Internet of Things Journal, vol. 1, no. 5, 2014.

[14] S. Guoqiang, C. Yanming, Z. Chao, Z. Yanxu, “Design and Implementation of a Smart IoT Gateway”, IEEE Int. Conf. Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, 2013.

[15] S.K. Datta, C. Bonnet, "Smart M2M Gateway Based Architecture for M2M Device and Endpoint Management", IEEE Int. Conf. Internet of Things, 2014.

[16] Kura. Available online at: https://eclipse.org/kura.

[17] MQTT - Message Queue Telemetry Transport. Available online at: http://mqtt.org.

[18] CoAP – Constrained Application Protocol. Available online at: http://coap.technology.

[19] Z. Shelby, K. Hartke, C. Bormann, “The Constrained Application Protocol (CoAP)”, 2014.

[20] M. Nottingham, E. Hammer-Lahav, “Defining Well-Known Uniform Resource Identifiers (URIs)”, IETF Internet Draft, 2010. Available online at: https://tools.ietf.org/html/rfc5785.

[21] M. Nottingham, E. Hammer-Lahav, “Defining Well-Known Uniform Resource Identifiers (URIs)”, IETF Internet Draft, 2010. Available online at: https://tools.ietf.org/html/rfc5785.

[22] M. Nottingham, “Web Linking”, IETF Internet Draft, 2010. Available online at: https://tools.ietf.org/html/rfc5988.

[23] Z. Shelby, “Constrained RESTful Environments (CoRE) Link Format”, IETF Internet Draft, 2012. Available online at: https://tools.ietf.org/html/rfc6690.

[24] Mosquitto. Available oline at: http://mosquitto.org.

978-1-5090-1131-5/16/$31.00 ©2016 IEEE